DAA Mid-1 Important Q & A

1) a) Difference between Algorithm and Psudeocode

Ans:-

Comparison Table Between Pseudocode and Algorithm

Parameters of comparison	Pseudocode	Algorithm
Definition	A "text-based" tool useful in developing algorithm	A sequential set of orders to complete certain task in a program
Aim	To simplify the programming language so that humans can understand without having prior knowledge about programming language	To help in performing the task and get the desired output through defined steps
Characteristics	Clear beginning and end, usage of named variables and identifiers	Clear, unambiguous, defined input and output, language-independent and feasible
Advantages	Use of simple English language, designs the entire flow of the program, and can be easily converted to actual programming code	Step-wise representation which is simple and easy to understand and executes on available resources
Disadvantages	It cannot be compiled or executed and every designer has a different style of writing pseudocode	Time-consuming and certain branch and loop statements are difficult to depict in algorithm

1) b) Define Time and Space complexities.

Ans:-

(i) Space complexity :- Space complexity can be defined as how much space or memory required by an algorithm to run.

To compute the space complexity we have to use 2 factors i.e "constant" and "instance characteristics".

⇒ The space requirement "S (P)" can be given as $\boxed{S(P) = C + S_p}$

where c = constant i.e., fixed part and denotes space of inputs and outputs. This space is an amount of space taken by instructions, variables and identifiers. Where S_p = Space dependent upon instance characteristics.

⇒ Time complexity :- The time complexity can be defined as how much time required to execute an algorithm is called time complexity. There are two types for evaluating time complexity i.e compile time and run time. The time complexity is generally computed at run time or execution time

2) Explain Asymptotic Notations with Examples

Ans:-

→ Asymptotic Notations :- To choose the best algorithm, we need to check efficiency of each algorithm, the efficiency can be measured by computing time complexity of each algorithm. In this we have to implement different types of asymptotic notations they are:

(i) Big oh Notation (O)

(ii) Omega Notation (Ω)

(iii) Theta Notation (Θ)

(iv) Little oh Notation (o)

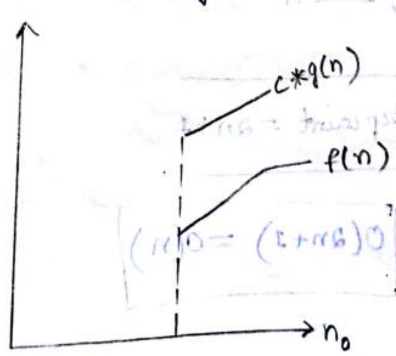
(v) Little omega Notation (ω).

(i) Big oh Notation :- Let $f(n)$ and $g(n)$ are any non negative function iff their exists positive constants 'c' and 'n_0' then the big oh notation can represent $\boxed{f(n) \leq c\,g(n), \quad n \geq n_0}$

→ In other words $f(n) \leq g(n)$ and $g(n)$ is some multiple of some constant 'c'.

⇒ Big oh Notation is denoted by "O". And it is the method of representing upper bound of algorithm.

⇒ Graph :-



Ex :-(i) $F(n) = 2n^2+3n+1$.

$F(n) \leq cg(n) \quad n \geq n_0$

$2n^2+3n+1 \leq 1 \rightarrow$ False

$2n^2+3n+1 \leq 3n \rightarrow$ False.

$2n^2+3n+1 \leq 2n^2 \rightarrow$ False.

$2n^2+3n+1 \leq 3n^2 \rightarrow n \geq n_0$

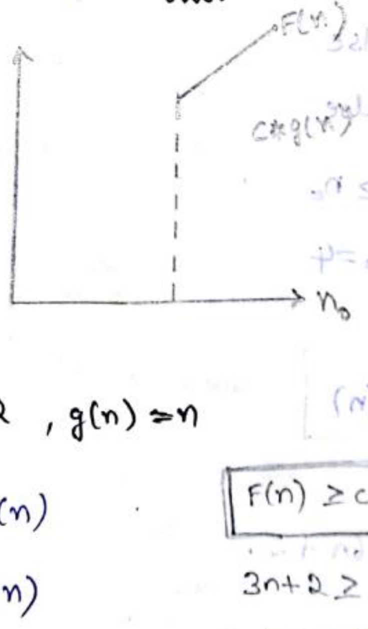
$45 \leq 48 \rightarrow n_0 = 4$

$\therefore c=3, \ n_0 = 4$

$\therefore \boxed{O(2n^2+3n+1) = O(n^2)}$

(ii) Omega Notation (Ω) :- Let $f(n)$ and $g(n)$ are any non negative functions. if their exists a positive constants 'c' and 'n_0' then the omega notation can be represented as $\boxed{F(n) \geq cg(n)}$ and it is represented by the symbol "Ω".

⇒ Graph :-



Ex :-(i) $F(n) = 3n+2$, $g(n) = n$

$F(n) \geq cg(n)$

$3n+2 \geq c(n)$

Sub $n=1 \Rightarrow 5 \geq c(1)$

$c=1. \quad 5 \geq 1$

$\boxed{F(n) \geq cg(n)}$

$3n+2 \geq 3n$

put $n=1 \rightarrow 5 \geq 3$ (True)

$3n+2 \geq 3n \ \forall \ n \geq 3$

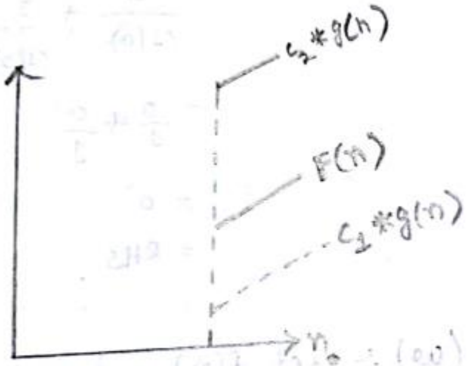
$c=3, \ g(n) = n, \ n_0 = 3$

$\boxed{3n+2 = \Omega(n)}$

(iii) Theta Notation (Θ) :- Let $f(n)$ and $g(n)$ are any non negative functions iff their exists c_1, c_2 and n_0 are the constants such that the theta notation can be represented as

$\boxed{c_1 * g(n) \le F(n) \le c_2 * g(n).}$ and it is represented by the Symbol "Θ"

$\Rightarrow$ Graph :-



Ex:- (i) $F(n) = 3n+2$, $g(n) = n$.

$c_1 * g(n) \le F(n) \le c_2 * g(n)$

$c_1 * n \le 3n+2 \le c_2 * n$

Sub $c_1 = 1$, $c_2 = 4$, $n = 2$

$1 * n \le 8 \le 4 * n$

$2 \le 8 \le 8$.

(or)

$3n \le 3n+2 \le 4n$

$n = 1 \Rightarrow 3 \le 5 \le 4$

$n = 2 \Rightarrow 6 \le 8 \le 8$ (T)

$3n \le 3n+2 \le 4n$, $n \ge 2$

$c_1 = 3$, $c_2 = 4$, $g(n) = n$, $n_0 = 2$

$\boxed{f(n) = \Theta g(n)}$

$\boxed{3n+2 = \Theta(n)}$

(iv) Little oh Notation :- (o) Let $f(n)$ and $g(n)$ are any two non negative functions if their exists $\boxed{\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0}$. In other words it can be written as $\boxed{F(n) = o(g(n))}$.

(i) $F(n) = 3n+2$, $g(n) = n^2$.

$$LHS := \lim_{n\to\infty} \frac{f(n)}{g(n)}$$

$$= \lim_{n\to\infty} \frac{3n+2}{n^2}$$

$$= \lim_{n\to\infty} \frac{3n}{n^2} + \frac{2}{n^2} = \lim_{n\to\infty} \frac{3}{n} + \frac{2}{n^2} = \frac{3}{\infty} + \frac{2}{\infty}$$

$$= \frac{3}{(1/0)} + \frac{2}{(1/0)}$$

$$= \frac{0}{1} + \frac{0}{1}$$

$$= 0$$

$$= RHS.$$

(v) Little omega Notation (ω) :- Let $f(n)$ and $g(n)$ are any two non negative functions if their exists: $\boxed{\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0}$

Ex :- $F(n) = 3n+2$, $g(n) = n^2$,

$$LHS := \lim_{n\to\infty} \frac{g(n)}{f(n)}$$

$$= \lim_{n\to\infty} \frac{n^2}{3n+2}$$

$$= \lim_{n\to\infty} \frac{n^{\cancel{2}}}{\cancel{n}(3+\frac{2}{n})} = \lim_{n\to\infty} \frac{n}{(3+\frac{2}{n})}$$

$$= \frac{\infty}{3+\frac{2}{\infty}} = 0 = RHS$$

3) **Solve following Recurrence Relation using Master's Theorem**

T(n) = 4T(n/4) + T(n)

Ans:-

There is no exact answer but a similar answer is

$\Rightarrow T(n) = 2T(n/2) + 1$

compare with general term $T(n) = aT(n/b) + F(n)$

Here $a = 2, b = 2, d = 0$ $(a^0 = 1)$

$a > b^d = 2 > 2^0 = 2 > 1$ condition is satisfied then

$\theta(n^{\log_b a}) = \theta(n^{\log_2 2})$

$= \theta(n)$.

4) **Write the Merge Sort algorithm and Sort the elements**

62,71,72,80,82,60,52,51,42

Ans:-

Algorithm :-

```
Algorithm  Mergesort (low, high)
{
  if (n=1) then
    return;
  else {
    if (low<high) then
    mid := (low+high)/2 ;
    mergesort (low, mid);
    mergesort (mid+1, high);
    combine (low, mid, high);
  }
}.
```

Given elements are

A[

1	2	3	4	5	6	7	8	9
62	71	72	80	82	60	52	51	42

].

$$mid = \frac{low + high}{2} = \frac{9+1}{2} = 5$$

We will divide the array into sub-arrays i.e

A[1-5] and A[6-9]

Now consider sub array A[1-5] elements i.e.

1	2	3	4	5
62	71	72	80	82

$$mid = \frac{1+5}{2} = 3 \text{ i.e}$$

1	2	3	4	5
62	71	72	80	82

$$mid = \frac{1+3}{2} = 2 \text{ i.e}$$

1	2	3	4	5
62	71	72	80	82

$$mid = \frac{1+2}{2} = 1.$$

1	2	3	4	5
62	71	72	80	82

$$mid = \frac{4+5}{2} = 4 \text{ i.e}$$

1	2	3	4	5
62	71	72	80	82

Now every set contains only one element and combine A[1] and A[2] and sort them i.e

1	2	3	4	5
62	71	72	80	82

Now combine A[1-2] and A[3] and sort them i.e

1	2	3	4	5
62	71	72	80	82

Now combine A[1-3] and A[4] and sort them i.e

1	2	3	4	5
62	71	72	80	82

Now combine A[1-4] and A[5] and sort them i.e

1	2	3	4	5
62	71	72	80	82

Thus the list A[1-5] is sorted and now we consider right sub array A[6-9] i.e

6	7	8	9
60	52	51	42

$$mid = \frac{6+9}{2} = 7$$

6	7	8	9
60	52	51	42

$$mid = \frac{6+7}{2} = 6.$$

6	7	8	9
60	52	51	42

$$mid = \frac{8+9}{2} = 8 \text{ i.e}$$

6	7	8	9
60	52	51	42

Now every set contains only one element. Now combine A[6] and A[7] and sort them i.e

6	7	8	9
52	60	51	42

Now combine A[6-7] and A[8] and sort them i.e

6	7	8	9
51	52	60	42

Now combine A[6-8] and A[9] and sort them i.e

6	7	8	9
42	51	52	60

Now combine two sorted sub-arrays and sort them i.e A[1-5] and A[6-9]

1	2	3	4	5	6	7	8	9
42	51	52	60	62	71	72	80	82

5) Write Union & Find Algorithms with examples.

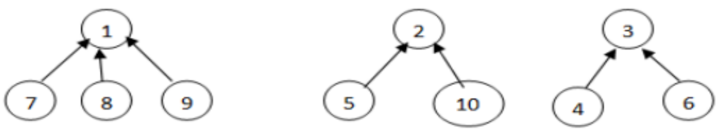
Ans:-

3 Union and Find Algorithms:

In presenting Union and Find algorithms, we ignore the set names and identify sets just by the roots of trees representing them. To represent the sets, we use an array of 1 to n elements where n is the maximum value among the elements of all sets. The index values represent the nodes (elements of set) and the entries represent the parent node. For the root value the entry will be '-1'.

Example:

For the following sets the array representation is as shown below.



S1 S2 S3

i	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
p	-1	-1	-1	3	2	3	1	1	1	2

1 Union Algorithm:

To perform union the **SimpleUnion(i,j)** function takes the inputs as the set roots i and j . And make the parent of i as j i.e, make the second root as the parent of first root.

```
Algorithm SimpleUnion(i,j)
        {
                P[i]:=j;
}
```

For example, let us consider an array. Initially parent array contains zero's.

	0	0	0	0	0	0	0	
Child←	1	2	3	4	5	6	7	↖parent

1) Union (1,3) → 1 ← 3

0	0	1	0	0	0	0
1	2	3	4	5	6	7

2) Union (2,5) → 1 ← 3

2 ← 5

0	0	1	0	2	0	0
1	2	3	4	5	6	7

3) Union (1,2) → 1 ← 3
↑
2 ← 5

0	0	1	0	2	0	0
1	2	3	4	5	6	7

Let us process the following sequence of union-find operations: →

Union (1,2); Union (2,3); Union (3,4);…………………………… ;Union (n-1,n);

i.e., Find(1), Find(2), Find(3),……………… Find(n).

This sequence results in the degenerate tree of diagram

1 → 2 → 3 → 4 →……………………………

Since the time taken for a union is constant, the n-1 union s can be processed in time O(n).

∴ Time complexity of union algorithm is O(n).

2 Find Algorithm:

The SimpleFind(i) algorithm takes the element i and finds the root node of i. It starts at I until it reaches a node with parent value -1.

Find (i) implies that if finds the root node of i^{th} node, in other words it returns the name of the set.

Eg:- union (1,3) → ①
↑
③

Find(3)=1 since its parent is 1 i.e., root node.

Algorithm:-

Algorithm find(i)

{

integer I,j;

while(parent (j)>0)

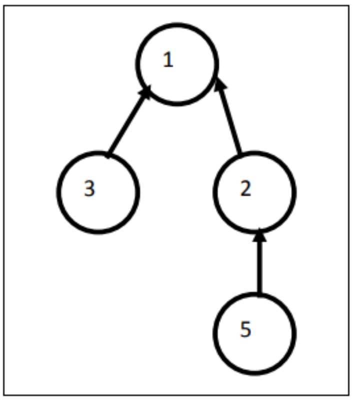
do j←parent(j)

repeat

return j;

}

EXAMPLE:



0	1	1	0	2	0
1	2	3	4	5	6

Find (5) j=5

While P(j)>0 that is, P(5)>0

⇨ 2>0 (true)
There fore j=2

While P(2) => 1>0 (true)

There fore j=1

While P(1) => 0>0 (false)

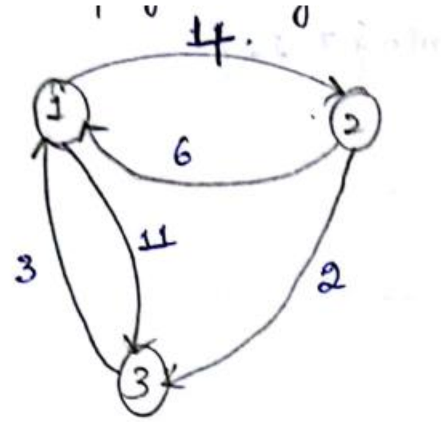
return j;

that is 1.

Therefore 1 is root node of node 5

The time complexity of find algorithm in nXn i.e $O(n^2)$

6) **Find All Pair Shortest Path Problem of Graph 'G' using Dynamic Programming**



Ans:-

Sol :- From the graph the cost adjacency matrix $A^0(i,j) = w[i,j] =$

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 11 \\ 2 & 6 & 0 & 2 \\ 3 & 3 & \infty & 0 \end{array}$$

∞ infinity becz 3, 2 no edge but adjacent.

our general formula is

$$\boxed{A^k(i,j) = \min\{A^{k-1}(i,j),\ A^{k-1}(i,k) + A^{k-1}(k,j)\}}$$

STEP 1 :- $k = 1$ i.e going from 'i' to 'j' through the intermediate vertex '1'. i.e $i = 1$.

$$A^1(\underset{i}{1},\underset{j}{1}) = \min\{A^0(1,1);\ A^0(1,1) + A^0(1,1)\}$$

$$= \min\{0,\ 0+0\}$$

$$\boxed{A^1(1,1) = 0}$$

$$A^1(\underset{i}{1},\underset{j}{2}) = \min\{A^0(1,2),\ A^0(1,1) + A^0(1,2)\}$$

$$= \min\{4,\ 0+4\}$$

$$= \min\{4,4\}$$

$$\boxed{A^1(1,2) = 4}$$

$$A^1(\underset{i}{1},\underset{j}{3}) = \min\{A^0(1,3),\ A^0(1,1) + A^0(1,3)\}$$

$$= \min\{11,\ 0+11\}$$

$$= \min\{11, 11\}$$

$$\boxed{A^1(1,3) = 11}$$

$$A^1(\underset{i}{2},\underset{j}{1}) = \min\{A^0(2,1),\ A^0(2,1) + A^0(1,1)\}$$

$$= \min\{6,\ 6+0\}$$

$$= \min\{6,6\}$$

$$\boxed{A^1(2,1) = 6}$$

$$A^1(\underset{i}{2},\underset{j}{2}) = \min\{A^0(2,2),\ A^0(2,1) + A^0(1,2)\}$$

$$= \min\{0,\ 6+4\}$$

$$= \min\{0, 10\}$$

$$\boxed{A^1(2,2) = 0}$$

$A^1(2,3) = \min\{A^0(2,3),\ A^0(2,1) + A^0(1,3)\}$

$= \min\{2,\ 6+11\}$

$= \min\{2, 17\}$

$\boxed{A^1(2,3) = 2}$

$A^1(3,1) = \min\{A^0(3,1),\ A^0(3,1) + A^0(1,1)\}$

$= \min\{3,\ 3+0\}$

$= \min\{3, 3\}$

$\boxed{A^1(3,1) = 3}$

$A^1(3,2) = \min\{A^0(3,2),\ A^0(3,1) + A^0(1,2)\}$

$= \min\{\infty,\ 3+4\} = \min\{\infty, 7\}$

Sc

$\boxed{A^1(3,2) = 7}$

$A^1(3,3) = \min\{A^0(3,3),\ A^0(3,1) + A^0(1,3)\}$

$= \min\{0,\ 3+11\}$

$= \min\{0, 14\}$

$\boxed{A^1(3,3) = 0}$

$$A^1 = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

STEP 2 :- Here $k=2$.

$A^2(1,1) = \min\{A^1(1,1),\ A^1(1,2) + A^1(2,1)\}$

$= \min\{0,\ 4+6\}$

$\boxed{A^2(1,1) = 0}$

$A^2(1,2) = \min\{A^1(1,2),\ A^1(1,2) + A^1(2,2)\}$

$= \min\{4,\ 4+0\}$

$\boxed{A^2(1,2) = 4}$

$A^2(1,3) = \min\{A^1(1,3),\ A^1(1,2) + A^1(2,3)\}$

$= \min\{11,\ 4+2\}$

$\boxed{A^2(1,3) = 6}$

$A^2(2,1) = \min\{A^1(2,1),\ A^1(2,2) + A^1(2,1)\}$

$= \min\{6,\ 0+6\}$

$\boxed{A^2(2,1) = 6}$

$A^2(2,2) = \min\{A^1(2,2),\ A^1(2,2) + A^1(2,2)\}$

$= \min\{0,\ 0+0\}$

$\boxed{A^2(2,2) = 0}$

$A^2(2,3) = \min\{A^1(2,3),\ A^1(2,2) + A^1(2,3)\}$

$= \min\{2,\ 0+2\}$

$\boxed{A^2(2,3) = 2}$

$A^2(3,1) = \min\{A^1(3,1),\ A^1(3,2) + A^1(2,1)\}$

$= \min\{3,\ 7+6\}$

$\boxed{A^2(3,1) = 3}$

$A^2(3,2) = \min\{A^1(3,2),\ A^1(3,2) + A^1(2,2)\}$

$= \min\{7,\ 7+0\}$

$\boxed{A^2(3,2) = 7}$

$A^2(3,3) = \min\{A^1(3,3),\ A^1(3,2) + A^1(2,3)\}$

$= \min\{0,\ 7+2\}$

$\boxed{A^2(3,3) = 0}$

STEP3 :- Here $k=3$

$$A^2 = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$A^3(1,1) = \min\{A^2(1,1),\ A^2(1,3) + A^2(3,1)\}$

$= \min\{0,\ 6+3\}$

$\boxed{A^3(1,1) = 0}$

Scann

$A^3(1,2) = \min\{A^2(1,2),\ A^2(1,3) + A^2(3,2)\}$

$= \min\{4,\ 6+7\}$

$= \min\{4, 13\}$

$\boxed{A^3(1,2) = 4}$

$A^3(1,3) = \min\{A^2(1,3),\ A^2(1,3) + A^2(3,3)\}$

$= \min\{6,\ 6+0\}$

$\boxed{A^3(1,3) = 6}$

$A^3(2,1) = \min\{A^2(2,1),\ A^2(2,3) + A^2(3,1)\}$

$= \min\{6,\ 2+3\}$

$= \min\{6,5\}$

$\boxed{A^3(2,1) = 5}$

$A^3(2,2) = \min\{A^2(2,2),\ A^2(2,3) + A^2(3,2)\}$

$= \min\{0,\ 2+7\}$

$= \min\{0,9\}$

$\boxed{A^3(2,2) = 0}$

$A^3(2,3) = \min\{A^2(2,3),\ A^2(2,3) + A^2(3,3)\}$

$= \min\{2,\ 2+0\}$

$\boxed{A^3(2,3) = 2}$

$A^3(3,1) = \min\{A^2(3,1),\ A^2(3,3) + A^2(3,1)\}$

$= \min\{3,\ 0+3\}$

$\boxed{A^3(3,1) = 3}$

$$A^3(3,2) = \min\{ \overset{i,j}{A^2(3,2)},\ \overset{i,k}{A^2(3,3)} + \overset{k,j}{A^2(3,2)} \}$$

$$= \min\{7,\ 0+7\}$$

$$\boxed{A^3(3,2) = 7}$$

$$A^3(3,3) = \min\{ \underset{i,j}{A^2(3,3)},\ \underset{i,k}{A^2(3,3)} + \underset{k,j}{A^2(3,3)} \}$$

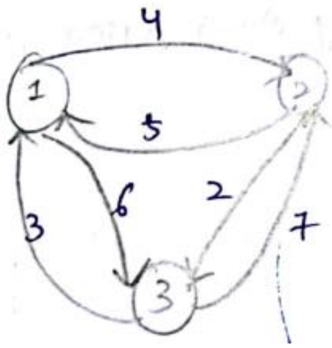
$$= \min\{0,\ 0+0\}$$

$$= \min\{0,0\}$$

$$\boxed{A^3(3,3) = 0}$$

$$A^3(i,j) = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$\Downarrow$



7) **Knapsack Problem using Dynamic Programming**

Ans:-

⇒ 0/1 knapsack problem by using dynamic programming :-

In olden days their was a store which contains different types of profits i.e $P_1, P_2, P_3 ---- P_n$ and cost $C_1, C_2, C_3 ---- C_n$ and weights $w_1, w_2, w_3 ---- w_n$ respectively. Now a thief wants to rob a store for that he brought a empty bag of size 'M'. Now his problem was in what way he can place the empty bag with maximum profit. This problem is called as knapsack problem. Here we have to use fractional values, 0's & 1's (binary no.'s)

Here knapsack means empty bag.

$\boxed{\text{Maximized } \sum p_i x_i \text{ subject to constraint } \sum w_i x_i \leq w}$

STEP 1 :- Initially compute $S^0 = \{(0,0)\}$ and ~~$S_1^i = \{(P,W) / [P=$~~

$$S_1^i = \{(P,W) / [(P-P_i),(W-W_i)] \in S^i\}$$

and S^{i+1} can be computed by merging S^i and S_1^i i.e

$$\boxed{S^{i+1} = S^i + S_1^i}$$

imp

STEP 2 :- Perging rule (Dominance rule) :-

If S^{i+1} contains P_j, W_j and (P_k, W_k) these two pairs satisfies the $P_j \le P_k$ and $W_j \ge W_k$ then we eliminate (P_j, W_j)

In perging rule basically the dominated tuples gets perged. In other words remove the pair with less profit and more weight

i.e $x_i = 1$ when $(P,W) \in S^i$ and $(P,W) \notin S^{i-1}$

$x_i = 0$ otherwise.

8) Strassen's Matrix Multiplication Time Complexity

Ans:-

Time complexity of strassen matrix :-

$$T(n) = 7T(n/2) + cn^2 \quad \text{--- (1)}$$

put $n = n/2$ in eq (1) we get

$$T(n/2) = 7T(n/2^2) + c(n/2)^2$$

sub $T(n/2)$ in eq (1) we get

$$T(n) = 7\left[7\,T(n/2^2) + c(n/2)^2\right] + cn^2$$

$$= 7^2 T(n/2^2) + (7/4)cn^2 + cn^2$$

$$= 7^3 T(n/2^3) + (7/4)^2 cn^2 + \left(\tfrac{7}{4}\right)cn^2 + cn^2$$

$$= 7^4 T(n/2^4) + \left[\left(\tfrac{7}{4}\right)^3 cn^2 + \left(\tfrac{7}{4}\right)^2 cn^2 + \left(\tfrac{7}{4}\right)cn^2 + cn^2\right]$$

$$\therefore \boxed{T(n) = 7^k\, T(n/2^k) + (7/4)^k \cdot cn^2}$$

put $n = 2^k$ i.e $k = \log_2^n$

$$T(n) = 7^{\log_2^n}\, T(2^k/2^k) + (7/4)^{\log_2^n} \cdot cn^2$$

$$= 7^{\log_2^n} \cdot T(1) + \frac{7^{\log_2^n}}{4^{\log_2^n}} \cdot cn^2$$

since $\boxed{a^{\log_c^b} = b^{\log_c^a}}$

$$T(n) = n^{\log_2^7} + \frac{n^{\log_2^7}}{\cancel{n^2}} \times c\cancel{n^2}$$

$$T(n) = n^{\log_2^7}\,[1 + c]$$

$$T(n) = O(n^{\log_2^7})$$

$$T(n) = O(n^{2.80})$$

$$\therefore \boxed{T(n) = O(n^{\log_2^7}) = O(n^{2.80})}$$

<u>All the best</u>

<u>Now do the</u>

<u>Rest…</u>